MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 1, 2018 / 2019

TLD 7011 – LOW-LEVEL DESIGN OF SOFTWARE

(All sections / Groups)

1 OCTOBER 2018
8:00 P.M. – 10:00 P.M.
(2 Hours)

INSTRUCTIONS TO STUDENTS

1. This Question paper consists of 4 pages (including the cover page) with 4 Questions only.

2. Attempt **ALL** questions. All questions carry equal marks and the distribution of the marks for each question is given.

3. Please write all your answers in the Answer Booklet provided.

QUESTION 1

(a) Describe the THREE characteristics of design patterns. [3 marks]

(b) Name TWO forms associated with design pattern. [2 marks]

(c) Based on the following scenario, explain why a *Prototype* design pattern would be suitable for this application. [2 marks]

> MyOffice Sdn Bhd is an office design company that is developing an application to help their designers to design office for their customers. A software tool is used to determine the furniture type (i.e. desk, storage and chair) to be installed in different rooms at the customers' offices. A key feature of the tool is the selection of the furniture type, where for each type of furniture there are a few options in terms of features such as colour and size.

(d) Draw the class model that implements the *Prototype* design pattern for the above application. [3 marks]

QUESTION 2

(a) *Interleaved concurrency* is a normal term used in parallel computing. With the help of a diagram, explain the term *interleaved concurrency*. [1 mark]

(b) Use `parallel.Invoke` to run the following four methods in parallel. [2 marks]

```
ConvertEllipse()
ConvertRectangle()
ConvertLines()
ConvertText()
```

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading.Tasks;

class Program
{
  static void Main(string[] args)
  {
       // write your code below this line

  }
}
```

Continue...

(c) Describe two decomposition techniques which can be used to decompose a task into various subtasks. [4 marks]

(d) Based on the following database, decompose the given query into subtasks (sub-table). Draw the task dependency graph. [3 marks]

MODEL = ``CIVIC'' AND YEAR = 2001 AND
(COLOR = ``GREEN'' OR COLOR = ``WHITE)

ID#	Model	Year	Color	Dealer	Price
4523	Civic	2002	Blue	MN	$18,000
3476	Corolla	1999	White	IL	$15,000
7623	Camry	2001	Green	NY	$21,000
9834	Prius	2001	Green	CA	$18,000
6734	Civic	2001	White	OR	$17,000
5342	Altima	2001	Green	FL	$19,000
3845	Maxima	2001	Blue	NY	$22,000
8354	Accord	2000	Green	VT	$18,000
4395	Civic	2001	Red	CA	$17,000
7352	Civic	2002	Red	WA	$18,000

QUESTION 3

(a) Given the following code, refactor it based on "replace temp with query" method. [3 marks]

```
double basePrice = _quantity * _itemprice;
if (basePrice > 1000)
  return basePrice * 0.95;
else
  return basePrice * 0.98;
```

(b) Describe two reasons why refactoring is important. [2 marks]

(c) Given the following code, refactor it so that we can move the private field `_interestRate` from class `Account` to class `AccountType` [4 marks]

Continue...

```
class Account {
  private AccountType _type;
  private double _interestRate;

  double interestForAmount_days (double amount, int days)
     {
       return _interestRate * amount * days / 365;
     }
}

class AccountType {

}
```

(d) Explain one condition when the "foreign method" refactoring method is suitable to be used. [1 mark]

QUESTION 4

(a) *"Reducing the lines of code in a high-level language improves the speed of the resulting machine code"*. State whether you agree with the above statement and justify your answer with example code. [3 marks]

(b) Given the following code, use *strength reduction* to tune the code. Rewrite the improved code. [2 marks]

```
for (i=0; i<n; i++)
   a[i] = i*conversion;
```

(c) Given the following code, how do you tune the code to make it faster. Rewrite the improved code. (Hint: $\log_{10} 2 = 0.69314718$) [2 marks]

```
unsigned int Log2(unsigned int x) {
   return (unsigned int) ( log (x) / log (2) );
}
```

(d) You observed the following statement in a program. Tune the code into a more efficient code by eliminating common subexpressions. [3 marks]

```
a = b + (c / d) - e * (c / d) + f * (d / c);
```

End of Question.